TITLE OF THE INVENTION

INFORMATION PROCESSING METHOD AND APPARATUS

FIELD OF THE INVENTION

This invention relates to an information processing method and apparatus and, more particularly, to an information processing technique for generating a compressed archive file by compressing a plurality of digital documents.

BACKGROUND OF THE INVENTION

In recent years, companies have adopted document management systems to convert paper documents into digital documents, and have promoted the re-use of these documents. In an early document management system, a paper document is scanned as an image by a scanner, and that image is registered and saved. Recently, however, digital documents created by personal computers have become widespread, and such digital documents can be registered and saved.

Also, in recent years, it is possible to extract arbitrary pages from a plurality of registered digital documents, to compress and archive these pages, and to bind these pages like a binder to form a single digital document (compressed archive file). Such files will also be referred to as digital binders hereinafter.

Such digital binders allow editing of internal files, and changing of individual files.

Conventional digital binders, however, suffer the following problems. In the conventional digital binder, in order to acquire the contents of the compressed digital binder, a series of processes for retrieving the entire digital binder onto a memory or as a file, and decompressing that binder are required. However, when data to be frequently accessed by the user is included in the digital binder, the user must decompress that data every time he or she accesses the data. For this reason, a very long time is required to process digital documents bound by the digital binder.

Upon processing the digital binder, a program must assure a large work memory area in advance on a main or sub storage device so as to decompress the compressed digital documents.

Such digital binders can be registered in a database that manages digital binders. The user can execute an edit process, save process, search process, and the like for digital binders registered in the database. One of the important functions of this database is a check-out/check-in function. In some cases, a plurality of users can access an arbitrary digital binder. In such environment, when a given user begins to user that digital binder, he or she can declare the right of edit to other users by setting the

digital binder in a check-out state using the check-out function. In this manner, using the check-out function, a single digital binder can be prevented from being opened and edited by a plurality of users at the same time. Upon completion of such edit process, the check-out state is canceled using the check-in function, and a change in digital binder made by the user is reflected on the digital binder on the database. After the check-out state is canceled, other users can edit that digital binder.

In such digital binder management method, however, when a given digital binder is checked out, that digital binder is copied, and the copy of the digital binder undergoes an edit process and the like.

Since the check-out process is made for each digital binder, all digital documents contained in each digital binder can be accessed by acquiring the copy of that digital binder, thus posing a security problem. Also, illicit copies are easily produced.

SUMMARY OF THE INVENTION

Accordingly, the present invention provides an information processing technique that archives digital documents after compression/non-compression is automatically selected for each digital document.

The exemplary method related to the present invention provides an information processing method for

generating an archive file that stores a plurality of digital documents, comprising: a checking step of checking based on a predetermined reference whether each digital document is to be stored in a compressed or non-compressed state; and a generation step of generating the archive file by controlling to store a digital document, which is determined in the checking step to be stored in a compressed state, in the compressed state, and controlling a digital document, which is determined in the checking step to be stored in a non-compressed state, in the non-compressed state.

Since compression/non-compression is determined based on a predetermined reference upon binding digital documents by a digital binder, the file size can be reduced while shortening the processing time required for expansion.

It is another object of the present invention to provide an information processing technique that can improve the security of a digital binder.

According to the present invention, the foregoing object is attained by providing, for example, an information processing method for generating an archive file that stores a plurality of digital documents, comprising: a checking step of checking based on a predetermined condition if each digital document main body is to be uploaded to a server; an abstract generation step of generating abstract data of the

digital document which is determined in the checking step to be uploaded; an upload step of uploading the digital document, which is determined in the checking step to be uploaded, to the server; and an archive file generation step of generating an archive file which stores the abstract data of the digital document which is determined in the checking step to be uploaded, and a digital document which is determined in the checking step not to be uploaded.

The invention is particularly advantageous since the security of the digital binder can be improved.

Other features and advantages of the present invention will be apparent from the following description taken in conjunction with the accompanying drawings, in which like reference characters designate the same or similar parts throughout the figures thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

Fig. 1 is a schematic view of a document management system;

Fig. 2 is a flow chart showing a digital binder generation process on the basis of the access frequencies of respective digital documents according to an embodiment of the present invention;

Fig. 3 is a flow chart showing a digital binder generation process on the basis of properties for respective formats of digital documents according to an embodiment of the present invention;

Fig. 4 is a flow chart showing an archive file generation process on the basis of the compression ratios of respective compressed digital documents according to an embodiment of the present invention;

Fig. 5 is a view showing an example of the internal configuration, and a compression/archive process and extraction/decompression process of a digital binder according to the present invention;

Fig. 6 is a schematic view showing a digital binder (archive file) according to an embodiment of the present invention, and digital documents bound by that binder;

Fig. 7 is a schematic view showing an example of the configuration of a digital binder according to the present invention;

Fig. 8 is a flow chart showing a process for generating a digital binder by determining based on a valid date whether each digital document is to be

stored in a digital binder or is to be uploaded to a server;

Fig. 9 is a flow chart upon printing digital documents in the digital binder generated in Fig. 8;

Fig. 10 is a flow chart showing a process for generating a digital binder by determining based on the file size of each digital document whether that digital document is to be stored in a digital binder or is to be uploaded to a server;

Fig. 11 is a flow chart upon downloading digital documents in the digital binder generated in Fig. 10; and

Fig. 12 is a block diagram showing the arrangement of an information processing apparatus which can be applied to the document management system according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Preferred embodiments of the present invention will now be described in detail in accordance with the accompanying drawings.

Fig. 1 is a schematic view of a digital binder (archive file) and digital documents bound by that binder according to an embodiment of the present invention. Digital documents 101 include various application files. The digital documents 101 are compressed to generate compressed digital documents 102,

and these documents 102 are archived to generate a digital binder 103. Various compression schemes may be used in compression. In this embodiment, compression is made using a predetermined reversible compression algorithm.

In Fig. 1, a file header 107 stores information of a digital binder itself, and information such as a date of creation and the like. A compression information table 108 stores information associated with the addresses of compressed digital documents 104 contained in the digital binder 103, and each compressed digital document 104 can be accessed with reference to this address.

In order to decompress each compressed digital document 104 bound by the digital binder 103 to a state in which an application can process that document, the document 104 is extracted from the digital binder 103 as a compressed digital document 105. The compressed digital document 105 undergoes a decompression process using a decompression algorithm corresponding to the compression scheme used in compression to generate a digital document 106.

(First Embodiment of Compression/non-compression Determination Process)

Fig. 2 is a flow chart showing a process for automatically determining compression/non-compression for each digital document on the basis of the access

frequencies of respective digital documents, and archiving the digital documents after a compression process and the like, upon compressing and archiving the digital documents.

In step 201, digital document number i is reset to zero to start a checking process from the first digital document.

In step 202, File_comp is obtained by dividing the access frequency of the i-th digital document by the total access frequency All_File_freq to calculate the ratio of accesses to file i to those to all files. When the value All_File_freq is smaller than a predetermined value, since it is statistically nonsense, File_comp is set to be "1".

It is checked in step 203 whether or not digital document i is frequently accessed, to see if File_comp > a predetermined threshold value threshold_comp. If it is determined that digital document i is frequently accessed, the flow advances to step 204. If it is determined that digital document i is accessed not so frequently, the flow advances to step 206.

It is checked in step 204 if a compression flag of digital document i is true or false, i.e., if that digital document has already been compressed. If digital document i has already been compressed, the flow advances to step 205 to decompress that document using a decompression algorithm corresponding to a

predetermined compression algorithm used in compression. If digital document i is not compressed, the flow jumps to step 208.

It is checked in step 206 if a compression flag of digital document i is true or false, i.e., if that digital document has already been compressed. If digital document i is not compressed, the flow advances to step 207 to compress that document according to the predetermined compression algorithm. If digital document i has already been compressed, the flow jumps to step 208.

In step 208, i is incremented by 1 to select the next document to be processed.

The current digital document number i and the total number filenum of files are compared in step 209 to check if all digital documents have been checked. If $i < filenum$, the flow returns to step 202 to check the next digital document. If $i \geq filenum$, i.e., if it is determined that all the digital documents have undergone the compression/non-compression determination process, the flow advances to step 210. In step 210, all the digital documents undergo an archive process (bind process) to generate a digital binder.

(Second Embodiment of Compression/non-compression Determination Process)

Fig. 3 is a flow chart showing a process for checking the format of each digital document with

reference to the extension of that digital document, automatically determining compression/non-compression for each digital document on the basis of the properties for respective digital documents, and archiving the digital documents, upon compressing and archiving the digital documents.

In step 301, digital document number i is reset to zero to start a checking process from the first digital document.

In step 302, File_comp represents the format type of digital document i. It is checked in step 302 if File_comp is a predetermined format (text format in this embodiment). Some digital documents have already been compressed in a unique format, and the compression efficiency cannot be improved if such digital documents are re-compressed. For this reason, it is checked if the type of digital document i is a specific type. That is, if digital document i is a non-compressed digital document or a digital document of a type which can be re-compressed to improve the compression efficiency, the flow advances to step 305. On the other hand, if digital document i is a digital document which cannot improve the compression efficiency if it is compressed, the flow advances to step 303.

It is checked in step 303 if a compression flag of digital document i is true or false, i.e., if that digital document has already been compressed. If

digital document i has already been compressed, the flow advances to step 304 to decompress that document using a decompression algorithm corresponding to a predetermined compression algorithm used in compression. If digital document i is not compressed, the flow jumps to step 307.

It is checked in step 305 if a compression flag of digital document i is true or false, i.e., if that digital document has already been compressed. If digital document i is not compressed, the flow advances to step 306 to compress that document according to the predetermined compression algorithm. If digital document i has already been compressed, the flow jumps to step 307.

In step 307, i is incremented by 1.

The current digital document number i and the total number filenum of files are compared in step 308 to check if all digital documents have been checked. If $i < filenum$, the flow returns to step 302 to check the next digital document. If $i \geq filenum$, i.e., if it is determined that all the digital documents have undergone the compression/non-compression determination process, the flow advances to step 309.

In step 309, all the digital documents undergo an archive process (bind process) to generate a digital binder.

(Third Embodiment of Compression/non-compression Determination Process)

Fig. 4 is a flow chart showing a process for automatically determining compression/non-compression for each digital document on the basis of the compression ratios of respective digital documents, and archiving the digital documents, upon compressing and archiving the digital documents.

In step 401, digital document number i is reset to zero to start a checking process from the first digital document.

It is checked in step 402 if a compression flag of digital document i is true or false, i.e., if that digital document has already been compressed. If it is determined that digital document i is not compressed, the flow advances to step 403 to compress that document according to a predetermined compression algorithm. If it is determined that digital document i has already been compressed, the flow jumps to step 404.

In step 404, File_comp is obtained by dividing the compressed size of digital document i by its non-compressed size to calculate the compression ratio of digital document i. As this value is smaller, the document of interest is a compressed digital document with a higher compression ratio.

It is checked in step 405 if File_comp > a predetermined threshold value threshold_comp. If

File_comp > threshold_comp, it is determined that digital document i has a low compression ratio, and the flow advances to step 406. Conversely, if digital document i has a high compression ratio, the flow advances to step 407.

In step 406, it is determined that the compression efficiency is not improved if document i is compressed, and an original digital document before compression is set as a file to be archived. The flow then advances to step 408.

In step 407, it is determined that digital document i has high compression efficiency, and a digital document after compression is set as a file to be archived. The flow then advances to step 408.

In step 408, i is incremented by 1. The current digital document number i and the total number filenum of files are compared in step 409 to check if all digital documents have been checked. If i < filenum, the flow returns to step 402 to check the next digital document. If i ≥ filenum, it is determined that all the digital documents have undergone the compression/non-compression determination process, and the flow advances to step 410.

In step 410, all the digital documents undergo an archive process (bind process) using digital documents (non-compressed documents) selected in step 406 and

those (compressed documents) selected in step 407 to generate a digital binder.

(Concept of Compression/non-compression Determination Process)

Fig. 5 is a schematic view showing a process for generating a digital binder 503 by archiving compressed digital documents 502 generated by compressing digital documents 501, and non-compressed digital documents 507 together upon archiving digital documents using the method of the present invention (step 210 in Fig. 2, step 309 in Fig. 3, step 410 in Fig. 4), and a process for extracting compressed digital documents 505 and non-compressed digital documents 509 from the digital binder 503.

A digital document 501 is determined to be archived in a compressed state when it is checked based on the above reference. The digital document 501 is compressed according to a predetermined compression algorithm to generate a compressed digital document 502. A digital document 507 is determined not to be compressed when it is checked based on the above reference. Digital documents including these compressed/non-compressed documents together are archived to generate a digital binder 503. Upon archiving, information associated with each digital document (address information indicating the location of each digital document, size information, information

indicating compression/non-compression, and the like) is stored in a compression information table of the digital binder 503.

In order to process compressed digital documents 504 and non-compressed digital documents 508 in the digital binder 503 by an application, compressed digital documents 505 and non-compressed digital documents 509 are extracted. Each compressed digital document 505 is decompressed in accordance with a decompression algorithm corresponding to the compression algorithm used in compression to generate a digital document 506. When each non-compressed digital document 508 is processed, a non-compressed digital document 509 is temporarily extracted. Alternatively, the application can execute a process by directly accessing the start address of the non-compressed digital document 508 with reference to the compression information table.

As described above, according to the above embodiments, since compression/non-compression is determined based on a predetermined reference upon binding digital documents by a digital binder, the file size can be reduced while shortening the processing time required for decompression.

Also, according to the above embodiments, since an unnecessary compression/expansion process for digital documents bound by the digital binder is

skipped, the processing time of digital documents bound by the digital binder can be shortened.

Furthermore, according to the above embodiments, upon processing digital documents bound by the digital binder, since a process can be done by directly accessing each non-compressed digital document in the digital binder, the processing area to be assured on the main and sub storage devices can be saved.

(Embodiment About Improvement of Security)

Fig. 6 is a schematic view showing a digital binder (archive file) that can be handled by an information processing apparatus according to an embodiment of the present invention, and digital documents bound by that binder. The same reference numerals denote the already explained parts, and a description thereof will be omitted. A compression information table 611 stores information associated with the addresses of compressed digital documents 504 and non-compressed digital documents 508 included in a digital binder 503. With reference to these addresses, the compressed digital documents 504 and non-compressed digital documents 508 can be accessed. In order to decompress each compressed digital document 504 bound by the digital binder 503, it is extracted as a compressed digital document 505 from the digital binder 503. The compressed digital document 505 undergoes a decompression process using a decompression algorithm

corresponding to the compression scheme used in compression, thus generating a digital document 506.

In case of the non-compressed document 508, a digital document 509 can be extracted by acquiring its file offset and file size. Abstract text data (index data) 610 is generated using text data of each digital document 501. Since this data indicates the contents of digital documents included in the digital binder 503, when the digital documents 504 and 508 have been updated, it can be updated while reflecting their contents.

The information processing apparatus of this embodiment can handle the digital binder shown in Fig. 6. Also, the present invention can achieve processes shown in Fig. 7. Fig. 7 is a schematic view showing a process for determining digital documents 702 whose main bodies are not to be archived using some condition to be described later, generating abstract data 703 of the digital documents 702 which are not to be archived, uploading the documents which are not to be archived to an external server 704 without being archived in a digital binder 706, and generating a digital binder 706 by compressing and archiving digital documents 701 which are not updated, and the abstract data 703, and a process for downloading digital documents 705 that have been uploaded to the external

server 704 to restore digital documents to an editable state.

Details of the above processes will be described below. Upon generating a digital binder 706 from digital documents 700, some documents included in the digital documents 700 are categorized on the basis of a predetermined condition into digital documents 701 which are compressed and archived to form the digital binder, and digital documents 702 which are not suited to be archived in the digital binder.

The digital documents 702 are uploaded to the external server 704, and are saved as uploaded digital documents 705. Also, abstract data 703 associated with the digital documents 702 to be uploaded are generated.

The digital binder 706 is formed by compressing and archiving the digital documents 701 and abstract data 703.

Normally, upon browsing a digital binder, if the binder stores a digital document, an application which can browse that digital document and is associated with it is launched. Likewise, when the binder includes abstract data, an application is similarly launched to display abstracts and outlines of uploaded digital documents. Digital documents included in the digital binder 706 can be edited, printed, or searched by launching the application. If the original digital document 702 has been uploaded and the binder has only

its abstract data 703, the digital document 705 must be downloaded from the external server 704 to execute an edit process, print process, search process, and the like.

When the digital document 705 is downloaded, processes such as an edit process and the like are allowed. However, the digital document 705 cannot often be downloaded depending on a condition set upon uploading or a state upon downloading (e.g., out of valid date). In such cases, only its abstract data 709 is displayed, and processes such as an edit process and the like are inhibited.

A practical example will be explained below.

<Process Associated with Digital Binder that Stores Digital Documents given with Valid Dates>

Generation and use of a digital binder which can improve security by limiting use of digital documents by giving valid dates to them will be explained. Note that a print valid date is used as the valid date, but other valid dates (e.g., a browse valid date and the like) may be used.

Fig. 8 is a flow chart showing a process for setting a print valid date for an arbitrary digital document on the basis of user's designation, generating abstract data of the document set with the print valid date so as to be archived in place of its original digital document, saving the digital document main body

set with the print valid date to an external server, and generating a digital binder by archiving digital documents for which no print valid date is set and the abstract data, upon compressing and archiving digital documents as a digital binder.

In step 801, digital document number i is reset to zero to start a checking process from the first digital document.

It is checked in step 802 if it is designated to set a print valid date for digital document [i]. If it is determined that a print valid date is to be set, the flow advances to step 803; otherwise, the flow advances to step 807.

In step 803, a print valid date is set for target digital document [i].

In step 804, abstract data of digital document [i] set with the print valid date is generated.

In step 805, a main body of digital document [i] set with the print valid date is uploaded to an external server.

In step 806, information indicating that digital document [i] is uploaded and its abstract data is generated (including storage address information (link information) of the upload destination and the like) is appended to the digital binder.

In step 807, the digital document which is determined in step 802 not to be set with any print

valid date is compressed using a predetermined compression algorithm.

In step 808, i is incremented by 1. The current digital document number i and the total number filenum of all digital files are compared in step 809 to check if all digital documents have been checked. If $i < filenum$, the flow returns to step 802 to repeat the process for the next digital document i. If $i \geq filenum$, i.e., if it is determined that the checking processes of all the digital documents are complete, the flow advances to step 810.

In step 810, the digital binder is generated by archiving digital documents which are not set with any print valid date and the abstract data of digital documents set with print valid dates, and writing related information of respective digital documents and that of abstract data in the header field.

Fig. 9 is a flow chart showing a process executed when the user decompresses and prints the digital binder generated by the digital binder generation process shown in Fig. 8.

In step 901, digital document number i is reset to zero to start a checking process from the first digital document.

It is checked in step 902 if a print valid date is set for target digital document i. If a print valid

date is set for digital document i, the flow advances to step 903; otherwise, the flow advances to step 906.

In step 903, an inquiry is sent to the external server to acquire the time from it so as to compare it with the print valid date of target digital document i.

It is checked in step 904 if the print valid date of target digital document i has expired. If the print valid date does not expire, the flow advances to step 905; otherwise, the flow jumps to step 908.

In step 905, target digital document i is downloaded from the external server.

On the other hand, if it is determined in step 902 that no print valid date is set for digital document i, it is determined that use of target digital document i is not limited. Then, compressed digital document i is extracted and separated from the digital binder in step 906. In step 907, compressed digital document i undergoes a decompression process using a decompression algorithm corresponding to the predetermined compression algorithm used in compression.

In step 908, i is incremented by 1.

In step 909, digital document i which is downloaded in step 905 or is decompressed in step 907 is printed. For a digital document which is determined in step 904 that its print valid date has expired, abstract data can be printed in place of original digital document i.

The current digital document number i and the total number filenum of files are compared in step 910 to check if all digital documents have been checked. If i < filenum, the flow returns to step 902 to check the next digital document i. If i ≥ filenum, it is determined that all the digital documents have undergone the compression/non-compression determination process, thus ending the process.

As has been explained using Figs. 8 and 9, since a valid date is set for use of each digital document, the chances of inadvertent downloading of digital documents can be reduced, thus improving security.

<Process Associated with Digital Binder Given with File Size Limitation>

Fig. 10 is a flow chart showing a process for, when an administrator or the like sets a file size limitation of a digital document that can be archived by the user, and digital documents are to be compressed and archived as a digital binder, generating abstract data of a digital document with a file size larger than the limitation so as to be archived in place of an original digital document, saving that digital document main body in an external server, and generating a digital binder by archiving the abstract data and digital documents with sizes equal to or smaller than the file size limitation.

In step 1001, digital document number i is reset to zero to start a checking process from the first digital document.

In step 1002, target digital document i is compressed using a predetermined compression algorithm to generate temporary compressed digital document i.

In step 1003, file size Size_temp[i] of temporary compressed digital document i generated in step 1002 is compared with threshold value T_size of a file size which is set in advance by the administrator. If Size_temp[i] > T_size, it is determined that digital document i is too large to browse to recognize its contents, and the flow advances to step 1004. If Size_temp[i] ≤ T_size, it is determined that digital document i is allowed to browse, and the flow jumps to step 1007.

In step 1004, abstract data of digital document i which is determined in step 1003 to have too large a file size is generated.

In step 1005, a main body of digital document i is uploaded to the external server.

In step 1006, upload information (including storage address information of the upload destination and the like) indicating that digital document i is uploaded and its abstract data is generated is appended to the digital binder.

In step 1007, i is incremented by 1. The current digital document number i and the total number filenum of all digital files are compared in step 1008 to check if all digital documents have been checked. If $i < filenum$, the flow returns to step 1002 to execute a process for the next digital document i. If $i \geq filenum$, it is determined that the checking processes for all the digital documents are complete, and the flow advances to step 1009.

In step 1009, a digital binder is generated by archiving compressed digital documents which have compressed file sizes equal to or smaller than the predetermined threshold value, and abstract data of digital documents larger than a predetermined file size, which are generated in step 1004.

Fig. 11 is a flow chart showing a process executed when the user decompresses and prints the digital binder generated by the digital binder generation process shown in Fig. 10.

In step 1101, digital document number i is reset to zero to start a checking process from the first digital document.

It is checked in step 1102 if upload information is registered in the digital binder in association with digital document i. If upload information is registered, the flow advances to step 1103. If no upload information is registered, it is determined that

the digital binder includes the compressed digital document, and the flow advances to step 1106.

In step 1103, information (abstract or the like) associated with the uploaded file is presented to the user on the basis of the registered upload information.

It is confirmed in step 1104 if the user wants to download the presented file. If the user designates to download that file, the flow advances to step 1105 to download digital document i. On the other hand, if the user designates not to download that file, the flow jumps to step 1108.

In step 1106, archived compressed digital document i is extracted from the digital binder. In step 1107, compressed digital document i is decompressed to generate digital document i.

In step 1108, the digital document number is incremented by 1. The current digital document number i and the total number filenum of all digital files are compared in step 1109 to check if all digital documents have been checked. If $i < filenum$, the flow returns to step 1102 to repeat a process for the next digital document i. If $i \geq filenum$, the checking processes for all the digital documents are complete.

As has been explained using Figs. 10 and 11, since the file sizes of digital documents to be stored in the digital binder are limited, the file size of the

digital binder itself can be prevented from becoming too large, and the digital binder can be easily handled.

As described above, according to the above embodiments, the security of the digital binder can be improved.

Also, according to the above embodiments, the file size of the digital binder can be prevented from becoming inadvertently large. Hence, the digital binder can be easily handled.

Furthermore, according to the above embodiments, a digital binder that includes unwanted digital documents can be prevented from being transmitted.

Moreover, according to the above embodiments, upon using an uploaded digital document, since the user can use the latest digital document which is uploaded to the server independently of the acquisition date and time of the digital binder, he or she need not re-acquire the digital binder, thus allowing easy maintenance.

<Arrangement of Information Processing Apparatus>

Fig. 12 is a block diagram showing the arrangement of an information processing apparatus which can be applied to the document management system of the present invention. Note that the arrangement of the external server can adopt the same hardware arrangement as that shown in Fig. 12 except for the

software configuration associated with the aforementioned processes.

Referring to Fig. 12, a CPU 1202 controls a whole information processing apparatus 1201 via a main bus 1207, and also an input device 1211 (e.g., an image scanner, a storage device, another information processing apparatus connected via a network, a facsimile connected via a telephone line, or the like) externally connected to the information processing apparatus 1201 via an input I/F (interface) 1205. Furthermore, the CPU 1202 controls an output device (e.g., a printer, monitor, another information processing apparatus connected via a network, a facsimile connected via a telephone line, or the like) externally connected to the information processing apparatus 1201 via an output I/F 1206. The CPU 1202 executes an image input process, image process, color conversion process, image output control, and the like in accordance with instructions input from an input unit (e.g., a keyboard 1213, pointing device 1214, and pen 1215) via a KBD I/F (keyboard interface) 1208. Moreover, the CPU 1202 controls a display unit 1210 that displays image data input by the input device 1211 or image data generated using the keyboard 1213, pointing device 1214, and pen 1215 via a video I/F 1209.

A ROM 1203 stores various control programs that make the CPU 1202 execute various kinds of control.

These control programs may be stored in a hard disk drive (not shown). The control programs include a program corresponding to a flow chart of any one of Figs. 2 to 4 and Figs. 8 to 11.

On a RAM 1204, the CPU 1202 loads and executes an OS and other control program including those which are required to implement the present invention. Also, the RAM 1204 serves as various work areas and a temporary save area, which are used to execute the control programs. Furthermore, a VRAM (not shown) that temporarily holds image data input by the input device 1211 or image data generated using the keyboard 1213, pointing device 1214, and pen 1215 is assured on the RAM 1204.

<Other Embodiments>

Note that the present invention can be applied to an apparatus comprising a single device or to system constituted by a plurality of devices.

Furthermore, the invention can be implemented by supplying a software program, which implements the functions of the foregoing embodiments, directly or indirectly to a system or apparatus, reading the supplied program code with a computer of the system or apparatus, and then executing the program code. In this case, so long as the system or apparatus has the

functions of the program, the mode of implementation need not rely upon a program.

Accordingly, since the functions of the present invention are implemented by computer, the program code itself installed in the computer also implements the present invention. In other words, the claims of the present invention also cover a computer program for the purpose of implementing the functions of the present invention.

In this case, so long as the system or apparatus has the functions of the program, the program may be executed in any form, e.g., as object code, a program executed by an interpreter, or scrip data supplied to an operating system.

Example of storage media that can be used for supplying the program are a floppy disk, a hard disk, an optical disk, a magneto-optical disk, a CD-ROM, a CD-R, a CD-RW, a magnetic tape, a non-volatile type memory card, a ROM, and a DVD (DVD-ROM and a DVD-R).

As for the method of supplying the program, a client computer can be connected to a website on the Internet using a browser of the client computer, and the computer program of the present invention or an automatically-installable compressed file of the program can be downloaded to a recording medium such as a hard disk. Further, the program of the present invention can be supplied by dividing the program code

constituting the program into a plurality of files and downloading the files from different websites. In other words, a WWW (World Wide Web) server that downloads, to multiple users, the program files that implement the functions of the present invention by computer is also covered by the claims of the present invention.

Further, it is also possible to encrypt and store the program of the present invention on a storage medium such as a CD-ROM, distribute the storage medium to users, allow users who meet certain requirements to download decryption key information from a website via the Internet, and allow these users to decrypt the encrypted program by using the key information, whereby the program is installed in the user computer.

Furthermore, besides the case where the aforesaid functions according to the embodiments are implemented by executing the read program by computer, an operating system or the like running on the computer may perform all or a part of the actual processing so that the functions of the foregoing embodiments can be implemented by this processing.

Furthermore, after the program read from the storage medium is written to a function expansion board inserted into the computer or to a memory provided in a function expansion unit connected to the computer, a CPU or the like mounted on the function expansion board

or function expansion unit performs all or a part of the actual processing so that the functions of the foregoing embodiments can be implemented by this processing.

As many apparently widely different embodiments of the present invention can be made without departing from the spirit and scope thereof, it is to be understood that the invention is not limited to the specific embodiments thereof except as defined in the appended claims.